# Turtlebot3 launch file

turtlebot3_ws/src/turtlebot3_simulation/turtlebot3_gazebo/launch/turtlebot3_world.launch.py

```python
def generate_launch_description():
    launch_file_dir = os.path.join(get_package_share_directory(
        'turtlebot3_gazebo'), 'launch')
    pkg_gazebo_ros = get_package_share_directory('gazebo_ros')

    use_sim_time = LaunchConfiguration('use_sim_time', default='true')
    x_pose = LaunchConfiguration('x_pose', default='-2.0')
    y_pose = LaunchConfiguration('y_pose', default='-0.5')

    world = os.path.join(
        get_package_share_directory('turtlebot3_gazebo'),
        'worlds',
        'turtlebot3_world.world'
    )

    gzserver_cmd = IncludeLaunchDescription(
        PythonLaunchDescriptionSource(
            os.path.join(pkg_gazebo_ros, 'launch', 'gzserver.launch.py')
        ),
        launch_arguments={'world': world}.items()
    )

    gzclient_cmd = IncludeLaunchDescription(
        PythonLaunchDescriptionSource(
            os.path.join(pkg_gazebo_ros, 'launch', 'gzclient.launch.py')
        )
    )
```

# 월드 파일

# 가제보 실행

#로봇 스테이트 퍼블리시 런치 파일 실행

```python
    robot_state_publisher_cmd = IncludeLaunchDescription(
        PythonLaunchDescriptionSource(
            os.path.join(launch_file_dir, 'robot_state_publisher.launch.py')
        ),
        launch_arguments={'use_sim_time': use_sim_time}.items()
    )

    spawn_turtlebot_cmd = IncludeLaunchDescription(
        PythonLaunchDescriptionSource(
            os.path.join(launch_file_dir, 'spawn_turtlebot3.launch.py')
        ),
        launch_arguments={
            'x_pose': x_pose,
            'y_pose': y_pose
        }.items()
    )

    ld = LaunchDescription()

    # Add the commands to the launch description
    ld.add_action(gzserver_cmd)
    ld.add_action(gzclient_cmd)
    ld.add_action(robot_state_publisher_cmd)
    ld.add_action(spawn_turtlebot_cmd)

    return ld
```

#로봇 스폰 런치 파일 실행

turtlebot3_ws/src/turtlebot3_simulation/turtlebot3_gazebo/launch/robot_state_publisher.launch.py

```python
def generate_launch_description():
    TURTLEBOT3_MODEL = os.environ['TURTLEBOT3_MODEL']

    use_sim_time = LaunchConfiguration('use_sim_time', default='false')
    urdf_file_name = 'turtlebot3_' + TURTLEBOT3_MODEL + '.urdf'

    print('urdf_file_name : {}'.format(urdf_file_name))

    urdf_path = os.path.join(
        get_package_share_directory('turtlebot3_gazebo'),
        'urdf',
        urdf_file_name)

    with open(urdf_path, 'r') as infp:
        robot_desc = infp.read()
```

# urdf 파일

```python
    return LaunchDescription([
        DeclareLaunchArgument(
            'use_sim_time',
            default_value='false',
            description='Use simulation (Gazebo) clock if true'),

        Node(
            package='robot_state_publisher',
            executable='robot_state_publisher',
            name='robot_state_publisher',
            output='screen',
            parameters=[{
                'use_sim_time': use_sim_time,
                'robot_description': robot_desc
            }],
        ),
    ])
```

#로봇 스테이트 퍼블리시 노드 실행

turtlebot3_ws/src/turtlebot3_simulation/turtlebot3_gazebo/launch/spawn_turtlebot3.launch.py

```python
def generate_launch_description():
    # Get the urdf file
    TURTLEBOT3_MODEL = os.environ['TURTLEBOT3_MODEL']
    model_folder = 'turtlebot3_' + TURTLEBOT3_MODEL
    urdf_path = os.path.join(
        get_package_share_directory('turtlebot3_gazebo'),
        'models',
        model_folder,
        'model.sdf'
    )

    # Launch configuration variables specific to simulation
    x_pose = LaunchConfiguration('x_pose', default='0.0')
    y_pose = LaunchConfiguration('y_pose', default='0.0')

    # Declare the launch arguments
    declare_x_position_cmd = DeclareLaunchArgument(
        'x_pose', default_value='0.0',
        description='Specify namespace of the robot')

    declare_y_position_cmd = DeclareLaunchArgument(
        'y_pose', default_value='0.0',
        description='Specify namespace of the robot')
```

# sdf 파일

```python
start_gazebo_ros_spawner_cmd = Node(
    package='gazebo_ros',
    executable='spawn_entity.py',
    arguments=[
        '-entity', TURTLEBOT3_MODEL,
        '-file', urdf_path,
        '-x', x_pose,
        '-y', y_pose,
        '-z', '0.01'
    ],
    output='screen',
)

ld = LaunchDescription()

# Declare the launch options
ld.add_action(declare_x_position_cmd)
ld.add_action(declare_y_position_cmd)

# Add any conditioned actions
ld.add_action(start_gazebo_ros_spawner_cmd)

return ld
```

#스폰 엔터티 노드 실행

# Turtlebot3_gazebo

Model.sdf

```
$ ros2 node list
/turtlebot3_diff_drive
/turtlebot3_imu
/turtlebot3_joint_state
/turtlebot3_laserscan
/camera_driver
```

```
<plugin name="turtlebot3_imu" filename="libgazebo_ros_imu_sensor.so">
  <ros>
    <!-- <namespace>/tb3</namespace> -->
    <remapping>~/out:=imu</remapping>
  </ros>
</plugin>
```

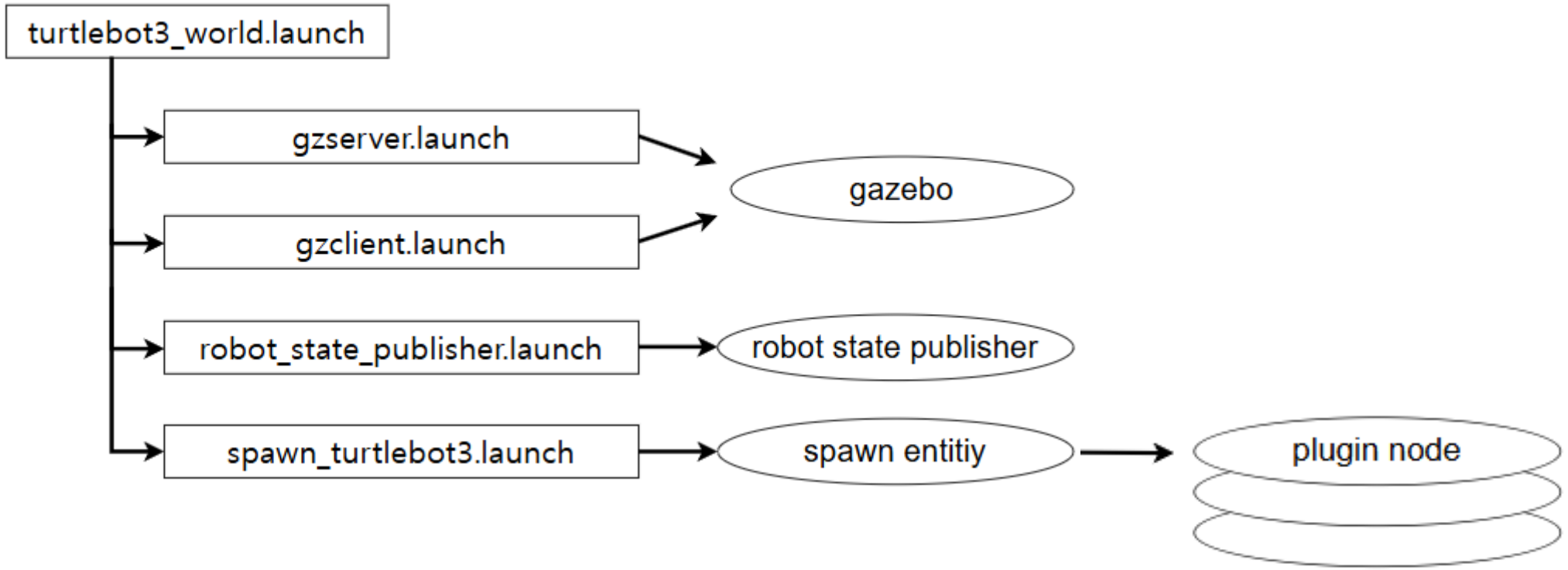
```
<plugin name="turtlebot3_laserscan" filename="libgazebo_ros_ray_sensor.so">
  <ros>
    <!-- <namespace>/tb3</namespace> -->
    <remapping>~/out:=scan</remapping>
  </ros>
```

```
<plugin name="turtlebot3_diff_drive" filename="libgazebo_ros_diff_drive.so">

  <ros>
    <!-- <namespace>/tb3</namespace> -->
  </ros>
```

```
<plugin name="camera_driver" filename="libgazebo_ros_camera.so">
  <ros>
    <!-- <namespace>test_cam</namespace> -->
    <!-- <remapping>image_raw:=image_demo</remapping> -->
    <!-- <remapping>camera_info:=camera_info_demo</remapping> -->
  </ros>
  <!-- camera_name>omit so it defaults to sensor name</camera_name-->
  <!-- frame_name>omit so it defaults to link name</frameName-->
  <!-- <hack_baseline>0.07</hack_baseline> -->
</plugin>
```

```
<plugin name="turtlebot3_joint_state" filename="libgazebo_ros_joint_state_publisher.so">
  <ros>
    <!-- <namespace>/tb3</namespace> -->
    <remapping>~/out:=joint_states</remapping>
  </ros>
  <update_rate>30</update_rate>
  <joint_name>wheel_left_joint</joint_name>
  <joint_name>wheel_right_joint</joint_name>
</plugin>
```


turtlebot3_world.launch
gzserver.launch
gzclient.launch
robot_state_publisher.launch
spawn_turtlebot3.launch
gazebo
robot state publisher
spawn entitiy
plugin node

turtlebot3_ws/src/turtlebot3/turtlebot3_navigation2/launch/navigation2.launch.py

```python
def generate_launch_description():
    use_sim_time = LaunchConfiguration('use_sim_time', default='false')
    map_dir = LaunchConfiguration(
        'map',
        default=os.path.join(
            get_package_share_directory('turtlebot3_navigation2'),
            'map',
            'map.yaml'))

    param_file_name = TURTLEBOT3_MODEL + '.yaml'
    param_dir = LaunchConfiguration(
        'params_file',
        default=os.path.join(
            get_package_share_directory('turtlebot3_navigation2'),
            'param',
            param_file_name))

    nav2_launch_file_dir = os.path.join(
        get_package_share_directory('nav2_bringup'), 'launch')

    rviz_config_dir = os.path.join(
        get_package_share_directory('nav2_bringup'),
        'rviz',
        'nav2_default_view.rviz')
```

# 파라미터 파일

```python
return LaunchDescription([
    DeclareLaunchArgument(
        'map',
        default_value=map_dir,
        description='Full path to map file to load'),
    DeclareLaunchArgument(
        'params_file',
        default_value=param_dir,
        description='Full path to param file to load'),
    DeclareLaunchArgument(
        'use_sim_time',
        default_value='false',
        description='Use simulation (Gazebo) clock if true'),
    IncludeLaunchDescription(
        PythonLaunchDescriptionSource([nav2_launch_file_dir, '/bringup_launch.py']),
        launch_arguments={
            'map': map_dir,
            'use_sim_time': use_sim_time,
            'params_file': param_dir}.items(),),
    Node(
        package='rviz2',
        executable='rviz2',
        name='rviz2',
        arguments=['-d', rviz_config_dir],
        parameters=[{'use_sim_time': use_sim_time}],
        output='screen'),
])
```

#nav2 브링업 런치 실행

#rviz 실행

/opt/ros/humble/share/nav2_bringup/launch/bringup_launch.py

```
remappings = [('/tf', 'tf'),
              ('/tf_static', 'tf_static')]
```

```
declare_slam_cmd = DeclareLaunchArgument(
    'slam',
    default_value='False',
    description='Whether run a SLAM')
```

```
Node(
    condition=IfCondition(use_composition),
    name='nav2_container',
    package='rclcpp_components',
    executable='component_container_isolated',
    parameters=[configured_params, {'autostart': autostart}],
    arguments=['--ros-args', '--log-level', log_level],
    remappings=remappings,
    output='screen'),
```

#컴포넌트 컨테이너 실행

```
IncludeLaunchDescription(
    PythonLaunchDescriptionSource(os.path.join(launch_dir, 'slam_launch.py')),
    condition=IfCondition(slam),
    launch_arguments={'namespace': namespace,
                      'use_sim_time': use_sim_time,
                      'autostart': autostart,
                      'use_respawn': use_respawn,
                      'params_file': params_file}.items()),
```

# slam = False -> 실행x

#슬램 런치 실행

/opt/ros/humble/share/nav2_bringup/launch/bringup_launch.py

#로컬라제이션 런치 실행

```
IncludeLaunchDescription(
    PythonLaunchDescriptionSource(os.path.join(launch_dir,
                                               'localization_launch.py')),
    condition=IfCondition(PythonExpression(['not ', slam])),
    launch_arguments={'namespace': namespace,
                      'map': map_yaml_file,
                      'use_sim_time': use_sim_time,
                      'autostart': autostart,
                      'params_file': params_file,
                      'use_composition': use_composition,
                      'use_respawn': use_respawn,
                      'container_name': 'nav2_container'}.items()),
```

#네비게이션 런치 실행

```
IncludeLaunchDescription(
    PythonLaunchDescriptionSource(os.path.join(launch_dir, 'navigation_launch.py')),
    launch_arguments={'namespace': namespace,
                      'use_sim_time': use_sim_time,
                      'autostart': autostart,
                      'params_file': params_file,
                      'use_composition': use_composition,
                      'use_respawn': use_respawn,
                      'container_name': 'nav2_container'}.items()),
```

/opt/ros/humble/share/nav2_bringup/launch/localization_launch.py

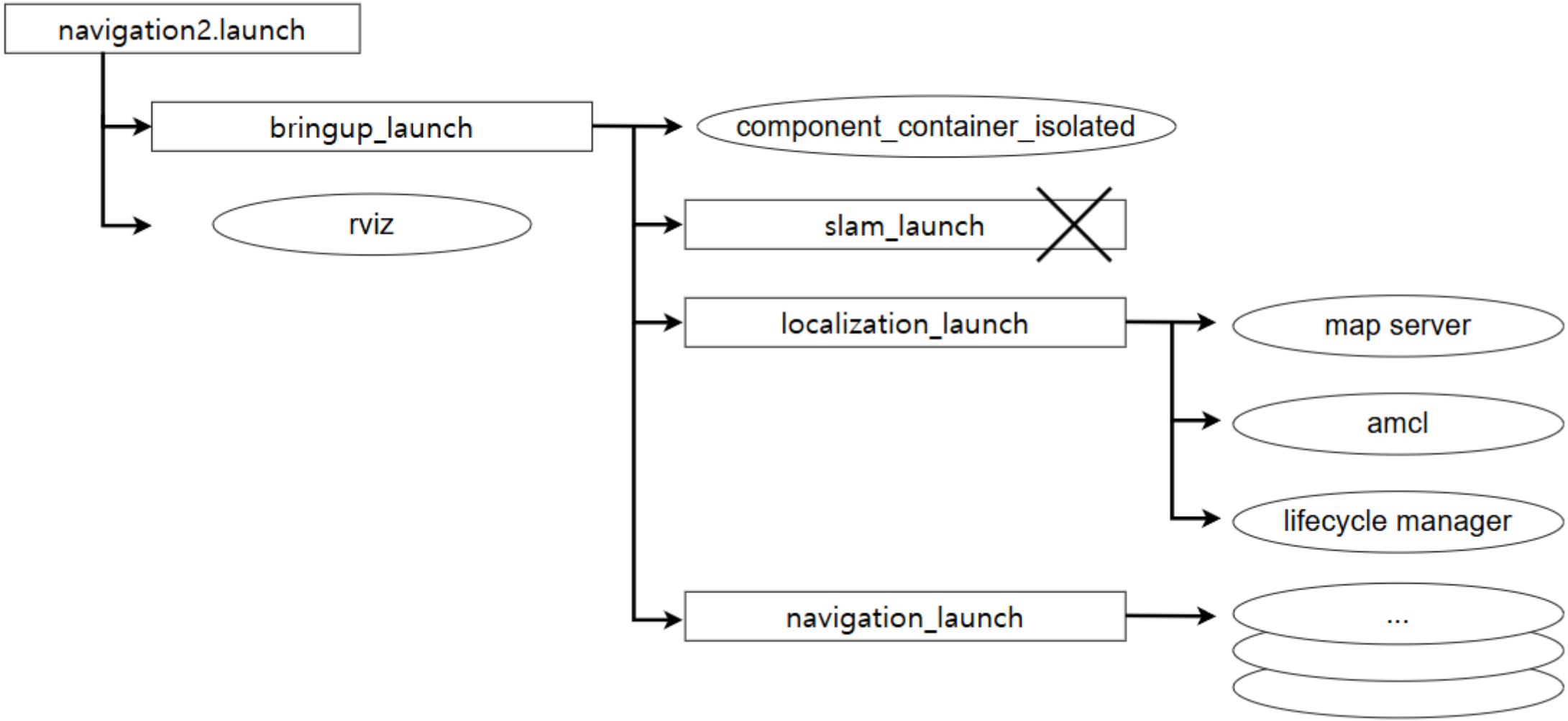
```
    Node(
        package='nav2_map_server',
        executable='map_server',
        name='map_server',
        output='screen',
        respawn=use_respawn,
        respawn_delay=2.0,
        parameters=[configured_params],
        arguments=['--ros-args', '--log-level', log_level],
        remappings=remappings),
    Node(
        package='nav2_amcl',
        executable='amcl',
        name='amcl',
        output='screen',
        respawn=use_respawn,
        respawn_delay=2.0,
        parameters=[configured_params],
        arguments=['--ros-args', '--log-level', log_level],
        remappings=remappings),
    Node(
        package='nav2_lifecycle_manager',
        executable='lifecycle_manager',
        name='lifecycle_manager_localization',
        output='screen',
        arguments=['--ros-args', '--log-level', log_level],
        parameters=[{'use_sim_time': use_sim_time},
                    {'autostart': autostart},
                    {'node_names': lifecycle_nodes}])
]
```

/opt/ros/humble/share/nav2_bringup/launch/navigation_launch.py

```
Node(
    package='nav2_controller',
    executable='controller_server',
    output='screen',
    respawn=use_respawn,
    respawn_delay=2.0,
    parameters=[configured_params],
    arguments=['--ros-args', '--log-level', log_level],
    remappings=remappings + [('cmd_vel', 'cmd_vel_nav')]),
Node(
    package='nav2_smoother',
    executable='smoother_server',
    name='smoother_server',
    output='screen',
    respawn=use_respawn,
    respawn_delay=2.0,
    parameters=[configured_params],
    arguments=['--ros-args', '--log-level', log_level],
    remappings=remappings),
Node(
    package='nav2_planner',
    executable='planner_server',
    name='planner_server',
    output='screen',
    respawn=use_respawn,
    respawn_delay=2.0,
    parameters=[configured_params],
    arguments=['--ros-args', '--log-level', log_level],
    remappings=remappings),
```

```
Node(
    package='nav2_behaviors',
    executable='behavior_server',
    name='behavior_server',
    output='screen',
    respawn=use_respawn,
    respawn_delay=2.0,
    parameters=[configured_params],
    arguments=['--ros-args', '--log-level', log_level],
    remappings=remappings),
Node(
    package='nav2_bt_navigator',
    executable='bt_navigator',
    name='bt_navigator',
    output='screen',
    respawn=use_respawn,
    respawn_delay=2.0,
    parameters=[configured_params],
    arguments=['--ros-args', '--log-level', log_level],
    remappings=remappings),
Node(
    package='nav2_waypoint_follower',
    executable='waypoint_follower',
    name='waypoint_follower',
    output='screen',
    respawn=use_respawn,
    respawn_delay=2.0,
    parameters=[configured_params],
    arguments=['--ros-args', '--log-level', log_level],
    remappings=remappings),
```

```
Node(
    package='nav2_velocity_smoother',
    executable='velocity_smoother',
    name='velocity_smoother',
    output='screen',
    respawn=use_respawn,
    respawn_delay=2.0,
    parameters=[configured_params],
    arguments=['--ros-args', '--log-level', log_level],
    remappings=remappings +
            [('cmd_vel', 'cmd_vel_nav'), ('cmd_vel_smoothed', 'cmd_vel')]),
Node(
    package='nav2_lifecycle_manager',
    executable='lifecycle_manager',
    name='lifecycle_manager_navigation',
    output='screen',
    arguments=['--ros-args', '--log-level', log_level],
    parameters=[{'use_sim_time': use_sim_time},
                {'autostart': autostart},
                {'node_names': lifecycle_nodes}]),
```


navigation2.launch
bringup_launch
rviz
component_container_isolated
slam_launch
localization_launch
map server
amcl
lifecycle manager
navigation_launch
...

# Namespace / remap

## namespace

ROS에서 namespace는 주로 **노드(node), 토픽(topic), 서비스(service)** 등을
**그룹화하고 구분**하기 위해 사용
서로 다른 로봇이나 시스템에서 발생할 수 있는 **이름 충돌을 방지**
**효율적**으로 **로봇 제어**

ros2 run turtlesim turtlesim_node –ros-args --remap __ns:=/robot1

## remapping

**Remapping**은 ROS에서 토픽, 서비스, 파라미터 등의 이름을 **동적으로 변경**

이름 충돌을 피하거나 시스템을 효율적으로 관리하는 데 사용

```
ros2 run turtlesim turtlesim_node –ros-args --remap turtle1/cmd_vel:=cmd_vel
```

## remapping

**Remapping**시 “ **/** “ 를 앞에 붙이면 글로벌화되어
네임스페이스의 영향을 받지 않음

ros2 run turtlesim turtlesim_node –ros-args --remap clear:=/clear –remap __ns:=/robot1

# Multi robot

# Turtlebot3_gazebo

ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py

```python
gzserver_cmd = IncludeLaunchDescription(
    PythonLaunchDescriptionSource(
        os.path.join(pkg_gazebo_ros, 'launch', 'gzserver.launch.py')
    ),
    launch_arguments={'world': world}.items()
)

gzclient_cmd = IncludeLaunchDescription(
    PythonLaunchDescriptionSource(
        os.path.join(pkg_gazebo_ros, 'launch', 'gzclient.launch.py')
    )
)

robot_state_publisher_cmd = IncludeLaunchDescription(
    PythonLaunchDescriptionSource(
        os.path.join(launch_file_dir, 'robot_state_publisher.launch.py')
    ),
    launch_arguments={'use_sim_time': use_sim_time}.items()
)

spawn_turtlebot_cmd = IncludeLaunchDescription(
    PythonLaunchDescriptionSource(
        os.path.join(launch_file_dir, 'spawn_turtlebot3.launch.py')
    ),
    launch_arguments={
        'x_pose': x_pose,
        'y_pose': y_pose
    }.items()
)
```

```
/gazebo
```

```
/robot_state_publisher
```

```
/turtlebot3_diff_drive
/turtlebot3_imu
/turtlebot3_joint_state
/turtlebot3_laserscan
/camera_driver
```

## Turtlebot3_gazebo

robot_state_publisher.launch.py

```
Node(
    package='robot_state_publisher',
    executable='robot_state_publisher',
    name='robot_state_publisher',
    output='screen',
    parameters=[{
        'use_sim_time': use_sim_time,
        'robot_description': robot_desc
    }],
),
```

```
remappings = [('/tf', 'tf'), ('/tf_static', 'tf_static')]
```

```
# Create state publisher node for that instance
turtlebot_state_publisher = Node(
    package='robot_state_publisher',
    namespace=namespace,
    executable='robot_state_publisher',
    output='screen',
    parameters=[{'use_sim_time': use_sim_time,
                 'publish_frequency': 10.0}],
    remappings=remappings,
    arguments=[urdf],
)
```

## Turtlebot3_gazebo

spawn_turtlebot3.launch.py

```
start_gazebo_ros_spawner_cmd = Node(
    package='gazebo_ros',
    executable='spawn_entity.py',
    arguments=[
        '-entity', TURTLEBOT3_MODEL,
        '-file', urdf_path,
        '-x', x_pose,
        '-y', y_pose,
        '-z', '0.01'
    ],
    output='screen',
)
```

```
# Create spawn call
spawn_turtlebot3_burger = Node(
    package='gazebo_ros',
    executable='spawn_entity.py',
    arguments=[
        '-file', os.path.join(turtlebot3_multi_robot,'models', 'turtlebot3_' + TURTLEBOT3_MODEL, 'model.sdf'),
        '-entity', robot['name'],
        '-robot_namespace', namespace,
        '-x', robot['x_pose'], '-y', robot['y_pose'],
        '-z', '0.01', '-Y', '0.0',
        '-unpause',
    ],
    output='screen',
)
```

# Turtlebot3_gazebo

Model.sdf

```
<?xml version="1.0" ?>
<sdf version="1.5">
  <model name="turtlebot3_waffle_pi">
  <pose>0.0 0.0 0.0 0.0 0.0 0.0</pose>

    <link name="base_footprint"/>

    <link name="base_link">

      <inertial>
        <pose>-0.064 0 0.048 0 0 0</pose>
        <inertia>
          <ixx>4.2111447e-02</ixx>
          <ixy>0</ixy>
          <ixz>0</ixz>
          <iyy>4.2111447e-02</iyy>
          <iyz>0</iyz>
          <izz>7.5254874e-02</izz>
        </inertia>
        <mass>1.3729096e+00</mass>
      </inertial>

      <collision name="base_collision">
        <pose>-0.064 0 0.048 0 0 0</pose>
        <geometry>
          <box>
            <size>0.265 0.265 0.089</size>
          </box>
        </geometry>
      </collision>

      <visual name="base_visual">
        <pose>-0.064 0 0 0 0 0</pose>
        <geometry>
          <mesh>
            <uri>model://turtlebot3_common/meshes/waffle_pi_base.dae</uri>
            <scale>0.001 0.001 0.001</scale>
```

# Turtlebot3_gazebo

```
<plugin name="turtlebot3_diff_drive" filename="libgazebo_ros_diff_drive.so">

  <ros>
    <!-- <namespace>/tb3</namespace> -->
  </ros>
```

```
<plugin name="turtlebot3_diff_drive" filename="libgazebo_ros_diff_drive.so">

  <ros>
    <!-- <namespace>/robot0</namespace> -->
    <remapping>/tf:=tf</remapping>
  </ros>
```

## Turtlebot3_navigate

navigation2.launch.py

```
PythonLaunchDescriptionSource([nav2_launch_file_dir, '/bringup_launch.py']),
launch_arguments={
    'map': map_dir,
    'use_sim_time': use_sim_time,
    'params_file': param_dir}.items(),
```

```
Node(
    package='rviz2',
    executable='rviz2',
    name='rviz2',
    arguments=['-d', rviz_config_dir],
    parameters=[{'use_sim_time': use_sim_time}],
    output='screen'),
```

bringup_launch.py

```
namespace = LaunchConfiguration('namespace')
```

```
IncludeLaunchDescription(
    PythonLaunchDescriptionSource(os.path.join(launch_dir,
                                               'localization_launch.py')),
    condition=IfCondition(PythonExpression(['not ', slam])),
    launch_arguments={'namespace': namespace,
                      'use_sim_time': use_sim_time,
                      'autostart': autostart,
                      'params_file': params_file,
                      'use_composition': use_composition,
                      'use_respawn': use_respawn,
                      'container_name': 'nav2_container'}.items()),

IncludeLaunchDescription(
    PythonLaunchDescriptionSource(os.path.join(launch_dir, 'navigation_launch.py')),
    launch_arguments={'namespace': namespace,
                      'use_sim_time': use_sim_time,
                      'autostart': autostart,
                      'params_file': params_file,
                      'use_composition': use_composition,
                      'use_respawn': use_respawn,
                      'container_name': 'nav2_container'}.items()),
```

## Turtlebot3_navigate

navigation2.launch.py

```
PythonLaunchDescriptionSource([nav2_launch_file_dir, '/bringup_launch.py']),
launch_arguments={
    'map': map_dir,
    'use_sim_time': use_sim_time,
    'params_file': param_dir}.items(),
```

```
Node(
    package='rviz2',
    executable='rviz2',
    name='rviz2',
    arguments=['-d', rviz_config_dir],
    parameters=[{'use_sim_time': use_sim_time}],
    output='screen'),
```

```
PythonLaunchDescriptionSource([nav2_launch_file_dir, '/bringup_launch.py']),
launch_arguments={
    'namespace': namespace,
    'use_namespace': 'True',
    'map': map_dir,
    'use_sim_time': use_sim_time,
    'params_file': param_dir}.items(),),
```

```
Node(
    package='rviz2',
    executable='rviz2',
    name='rviz2',
    namespace=namespace,
    arguments=['-d', rviz_config_dir],
    output='screen',
    remappings=[('/tf', 'tf'),
                ('/tf_static', 'tf_static'),
                ('/goal_pose', 'goal_pose'),
                ('/clicked_point', 'clicked_point'),
                ('/initialpose', 'initialpose')],
    parameters=[{'use_sim_time': use_sim_time}],
    output='screen'),
```

## Turtlebot3_navigate

navigation2.launch.py

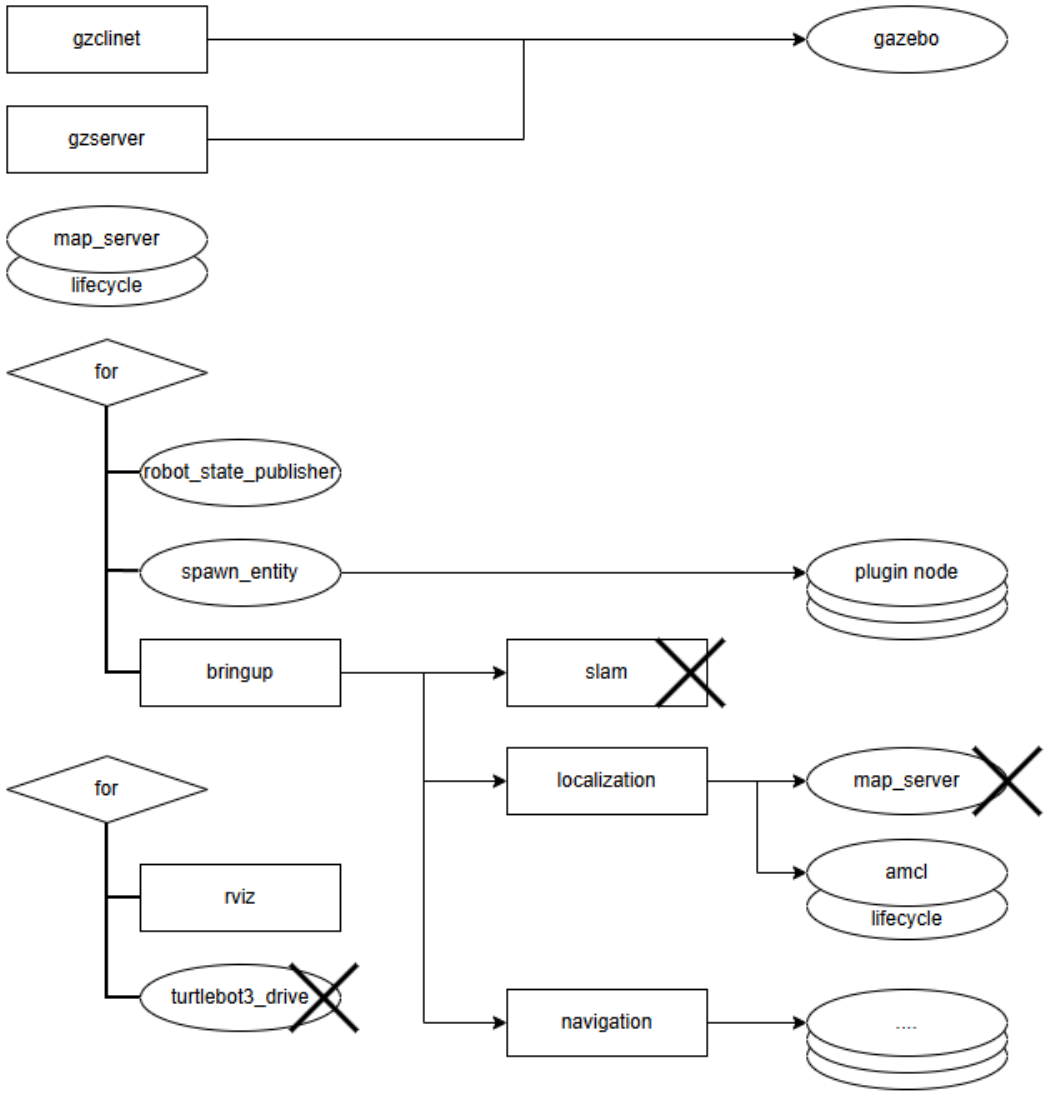
```
PythonLaunchDescriptionSource([nav2_launch_file_dir, '/bringup_launch.py']),
launch_arguments={
    'map': map_dir,
    'use_sim_time': use_sim_time,
    'params_file': param_dir}.items(),
```

```
Node(
    package='rviz2',
    executable='rviz2',
    name='rviz2',
    arguments=['-d', rviz_config_dir],
    parameters=[{'use_sim_time': use_sim_time}],
    output='screen'),
```

```
PythonLaunchDescriptionSource([nav2_launch_file_dir, '/bringup_launch.py']),
launch_arguments={
    'namespace': namespace,
    'use_namespace': 'True',
    'map': map_dir,
    'use_sim_time': use_sim_time,
    'params_file': param_dir}.items(),),
```

```
Node(
    package='rviz2',
    executable='rviz2',
    name='rviz2',
    namespace=namespace,
    arguments=['-d', rviz_config_dir],
    output='screen',
    remappings=[('/tf', 'tf'),
                ('/tf_static', 'tf_static'),
                ('/goal_pose', 'goal_pose'),
                ('/clicked_point', 'clicked_point'),
                ('/initialpose', 'initialpose')],
    parameters=[{'use_sim_time': use_sim_time}],
    output='screen'),
```

# Multi robot 2


gzclinet
gazebo
gzserver
map_server
lifecycle
for
robot_state_publisher
spawn_entity
plugin node
bringup
slam
for
localization
map_server
amcl
lifecycle
rviz
turtlebot3_drive
navigation
....

```python
def generate_launch_description():
    ld = LaunchDescription()

    # Names and poses of the robots
    robots = [
        {'name': 'tb1', 'x_pose': '-1.5', 'y_pose': '-0.5', 'z_pose': 0.01},
        {'name': 'tb2', 'x_pose': '-1.5', 'y_pose': '0.5', 'z_pose': 0.01},
        #{'name': 'tb3', 'x_pose': '1.5', 'y_pose': '-0.5', 'z_pose': 0.01},
        #{'name': 'tb4', 'x_pose': '1.5', 'y_pose': '0.5', 'z_pose': 0.01},
        # ...
        # ...
        ]

    TURTLEBOT3_MODEL = 'waffle'

    use_sim_time = LaunchConfiguration('use_sim_time', default='true')
    declare_use_sim_time = DeclareLaunchArgument(
        name='use_sim_time', default_value=use_sim_time, description='Use simulator time'
    )

    enable_drive = LaunchConfiguration('enable_drive', default='false')
    declare_enable_drive = DeclareLaunchArgument(
        name='enable_drive', default_value=enable_drive, description='Enable robot drive node'
    )

    enable_rviz = LaunchConfiguration('enable_rviz', default='true')
    declare_enable_rviz = DeclareLaunchArgument(
        name='enable_rviz', default_value=enable_rviz, description='Enable rviz launch'
    )


    turtlebot3_multi_robot = get_package_share_directory('turtlebot3_multi_robot')
```

# 사용할 로봇의 네임스페이스와 스폰할 위치
# 리스트 요소 수 만큼 로봇 스폰

```
urdf = os.path.join(
    turtlebot3_multi_robot, 'urdf', 'turtlebot3_' + TURTLEBOT3_MODEL + '.urdf'
)
```

```
world = os.path.join(
    get_package_share_directory('turtlebot3_multi_robot'),
    'worlds', 'empty_world.world')

gzserver_cmd = IncludeLaunchDescription(
    PythonLaunchDescriptionSource(
        os.path.join(get_package_share_directory('gazebo_ros'), 'launch', 'gzserver.launch.py')
    ),
    launch_arguments={'world': world}.items(),
)

gzclient_cmd = IncludeLaunchDescription(
    PythonLaunchDescriptionSource(
        os.path.join(get_package_share_directory('gazebo_ros'), 'launch', 'gzclient.launch.py')
    ),
)
```

# robot state publisher에 사용 될 urdf 파일 경로
# 여러 종류 로봇을 사용한다면 경로를 추가하고 밑의 for구문에 알맞게 적용

# 사용 할 world의 경로 지정

# 가제보 노드 실행

## Turtlebot3_navigate

```
map_server=Node(package='nav2_map_server',
    executable='map_server',
    name='map_server',
    output='screen',
    parameters=[{'yaml_filename': os.path.join(get_package_share_directory('turtlebot3_navigation2'), 'map', 'map.yaml'),
                },],
    remappings=remappings)

map_server_lifecyle=Node(package='nav2_lifecycle_manager',
        executable='lifecycle_manager',
        name='lifecycle_manager_map_server',
        output='screen',
        parameters=[{'use_sim_time': use_sim_time},
                    {'autostart': True},
                    {'node_names': ['map_server']}])
```

# localization launch의 map server 노드

# 작성한 맵 yaml파일 경로 지정

# 2개 이상의 맵 서버를 실행할 필요가 없으므로 따로 실행

# Turtlebot3_navigate

```
for robot in robots:

    namespace = [ '/' + robot['name'] ]

    # Create state publisher node for that instance
    turtlebot_state_publisher = Node(
        package='robot_state_publisher',
        namespace=namespace,
        executable='robot_state_publisher',
        output='screen',
        parameters=[{'use_sim_time': use_sim_time,
                     'publish_frequency': 10.0}],
        remappings=remappings,
        arguments=[urdf],
    )

    # Create spawn call
    spawn_turtlebot3_burger = Node(
        package='gazebo_ros',
        executable='spawn_entity.py',
        arguments=[
            '-file', os.path.join(
                turtlebot3_multi_robot,'models', 'turtlebot3_' + TURTLEBOT3_MODEL, 'model.sdf'),
            '-entity', robot['name'],
            '-robot_namespace', namespace,
            '-x', robot['x_pose'], '-y', robot['y_pose'],
            '-z', '0.01', '-Y', '0.0',
            '-unpause',
        ],
        output='screen',
```

# robots 리스트 요소 수 만큼 동작

## Turtlebot3_navigate

```
bringup_cmd = IncludeLaunchDescription(
        PythonLaunchDescriptionSource(
            os.path.join(nav_launch_dir, 'bringup_launch.py')),
            launch_arguments={
                                'slam': 'False',
                                'namespace': namespace,
                                'use_namespace': 'True',
                                'map': '',
                                'map_server': 'False',
                                'params_file': params_file,
                                'default_bt_xml_filename' : os.path.join(
                                    get_package_share_directory('nav2_bt_navigator'),
                                    'behavior_trees', 'navigate_w_replanning_and_recovery.xml'),
                                'autostart': 'true',
                                'use_sim_time': use_sim_time, 'log_level': 'warn'}.items()
                                )
```

# 앞에서 맵 서버 실행 했으므로 False

```python
if last_action is None:
    # Call add_action directly for the first robot to facilitate chain ir
    ld.add_action(turtlebot_state_publisher)
    ld.add_action(spawn_turtlebot3_burger)
    ld.add_action(bringup_cmd)

else:
    # Use RegisterEventHandler to ensure next robot creation happens only
    # Simply calling ld.add_action for spawn_entity introduces issues due
    spawn_turtlebot3_event = RegisterEventHandler(
        event_handler=OnProcessExit(
            target_action=last_action,
            on_exit=[spawn_turtlebot3_burger,
                     turtlebot_state_publisher,
                     bringup_cmd
                     ],
        )
    )

    ld.add_action(spawn_turtlebot3_event)
```

# 순차적으로 실행

```
initial_pose_cmd = ExecuteProcess(
    cmd=['ros2', 'topic', 'pub', '-t', '3', '--qos-reliability', 'reliable', namespace + ['/initialpose'],
        'geometry_msgs/PoseWithCovarianceStamped', message],
    output='screen'
)

rviz_cmd = IncludeLaunchDescription(
    PythonLaunchDescriptionSource(
        os.path.join(nav_launch_dir, 'rviz_launch.py')),
        launch_arguments={'use_sim_time': use_sim_time,
                          'namespace': namespace,
                          'use_namespace': 'True',
                          'rviz_config': rviz_config_file, 'log_level': 'warn'}.items(),
                           condition=IfCondition(enable_rviz)
                            )

drive_turtlebot3_burger = Node(
    package='turtlebot3_gazebo', executable='turtlebot3_drive',
    namespace=namespace, output='screen',
    condition=IfCondition(enable_drive),
)
```

# 2d pose estimate 토픽 발행

# rviz

# enable_drive = False이므로 실행x

# sdf 파일 플러그인 remapping 필요

```
<plugin name="turtlebot3_diff_drive" filename="libgazebo_ros_diff_drive.so">

  <ros>
    <!-- <namespace>/tb3</namespace> -->
  </ros>
```

```
<plugin name="turtlebot3_diff_drive" filename="libgazebo_ros_diff_drive.so">

  <ros>
    <!-- <namespace>/robot0</namespace> -->
    <remapping>/tf:=tf</remapping>
  </ros>
```

# 터틀봇3 관련 패키지에 종속되어 있으므로 터틀봇3 소싱 필요